

EdgeDebounce Version 1.2 (Mise à jour le 23 août 2017)

TUTORIEL

UN INTERRUPTEUR NE PEUT ÊTRE QUE FERMÉ OU OUVERT, NON?

Oui et non. Un interrupteur est conçu pour interrompre le courant ou lui laisser le passage.

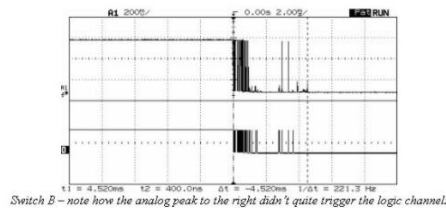
Habituellement, l'interrupteur est composé de deux (ou plus) pièces d'un élément conducteur. Quand les deux pièces sont en contact, l'interrupteur est fermé et laisse passer le courant entre les deux bornes.

Quand les deux pièces ne sont plus en contact, l'interrupteur est ouvert et le courant ne passe plus.



Malheureusement, il y a deux moments où l'interrupteur laisse passer le courant d'une manière aléatoire. C'est quand on l'ouvre ou quand on le ferme. Quand les contacts sont tout près l'un de l'autre, mais que l'interrupteur n'est pas tout à fait ouvert ou fermé. Les électrons peuvent migrer d'un contact vers l'autre de manière sporadique. On pourrait le comparer à des éclairs dans un orage. Plus les contacts sont près l'un de l'autre, plus il y aura d'éclairs laissant passer le courant de l'un à l'autre.

Voici un exemple, à l'oscilloscope, d'un interrupteur que l'on est en train de fermer¹ :

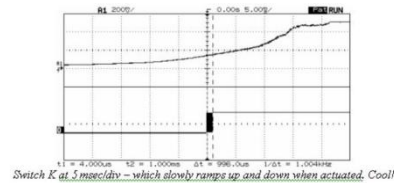


Switch B - note how the analog peak to the right didn't quite trigger the logic channel.

Dans le haut de l'image, on voit le voltage tel que mesuré. Dans le bas on voit le voltage retourné par un convertisseur analogique-digital. Un équivalent de `digitalRead()`. On peut voir les « éclairs », très fréquents au début, puis qui s'espacent jusqu'à ce que le convertisseur se stabilise à 0 V. Il se passe 2 millisecondes entre le moment où l'interrupteur commence à être relâché et celui où il est enfin stable. Si on met en boucle un `digitalRead()`

pendant cette période, il retournera environ 650 lectures et on ne peut être certain d'aucune d'elles.

Voici un autre exemple :



Switch K at 5 msec/div - which slowly ramps up and down when actuated. Cool!

Cet interrupteur ne se comporte pas comme l'autre. Le voltage augmente progressivement. Si on regarde l'image du bas, il se passe quelque chose d'étrange. Pendant un certain temps le convertisseur analogique-digital est stable à 0 V. Puis, il devient très instable, passant de 5 V à 0 V très rapidement. Il se stabilise enfin à 5V, quand l'interrupteur est effectivement fermé. Cela tient aux spécifications

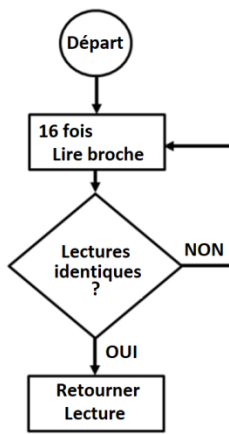
du convertisseur analogique-digital. S'il mesure entre 0 et 0,8 V sur la broche, il retourne LOW. S'il mesure entre 2,2 et 5 V, il retourne HIGH. Entre 0,8 et 2,2 V, il retourne au hasard HIGH ou LOW, ne nous laissant pas savoir ce qu'il fait.

¹ Ces exemples sont tirés de : <http://www.ganssle.com/debouncing.htm>

COMMENT SAVOIR SI L'INTERRUPTEUR EST RÉELLEMENT FERMÉ OU OUVERT?

Comme nous venons de le voir, le comportement erratique ne se produit que lorsque l'interrupteur est en voie de se fermer ou de s'ouvrir. L'approche en sera une de patience. Au lieu de ne prendre qu'une lecture de la broche, nous en prendrons plusieurs successivement. Supposons que l'on décide de prendre 16 lectures successives aussi rapprochées que possible (le tout se passe en environ 90 millièmes de secondes). On vérifie ensuite que toutes les lectures sont identiques (toutes HIGH ou toutes LOW). Si c'est le cas, les probabilités que le résultat soit valide est grand. Si, par contre, il y a des lectures HIGH parmi des LOW, on a la certitude que l'on est en phase de transition. Comme on l'a dit précédemment, il suffit d'être patient et d'attendre que la tempête cesse. On va reprendre des séries de 16 lectures jusqu'à ce qu'elles concordent toutes (16 HIGH ou 16 LOW).

L'ALGORITHME²



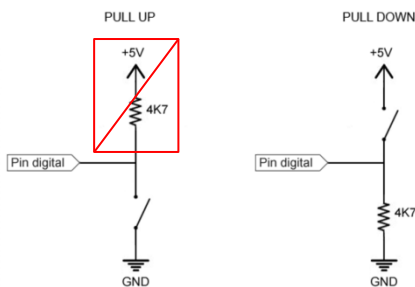
1. Lire 16 fois la broche
2. Si toutes les lectures sont identiques, retourner la lecture
3. Si non, reprendre à l'étape 1

Par défaut, les broches sont lues en rafales de 16. Il est toutefois possible de changer la taille de ces rafales entre 1 (aucune stabilisation du signal) et 32 (nombre maximal de lectures faites en rafale $\approx 180 \mu s$).

Pour la plupart des interrupteurs, 16 lectures par rafale sont suffisantes pour stabiliser le signal. Selon le type d'interrupteur ou de l'application qu'on désire en faire, il est possible d'en changer la valeur. Par exemple, j'ai plusieurs encodeurs et j'obtiens de meilleurs résultats avec ceux-ci avec des rafales de 8 lectures.

LE MONTAGE DE L'INTERRUPTEUR

Il existe deux manières de monter un interrupteur :



Sur l'image de gauche, en mode d'excursion haute (pull up), une des broches de l'interrupteur est reliée à la terre (GND) et l'autre à une des broches digitales de l'Arduino. La résistance montrée (entre +5V et la broche digitale) n'a pas à être placée. Arduino possède cette résistance à l'interne.

Dans un sketch, on initialise habituellement la broche de la manière suivante :

```
pinMode(2, INPUT_PULLUP)
```

En mode d'excursion haute, quand le contact est fermé, digitalRead() retournera LOW. Quand le contact est ouvert, il retournera HIGH.









Sur l'image de droite, en mode d'excursion basse (pull down), une des broches de l'interrupteur est reliée à +5 V, et l'autre à une des broches digitales de l'Arduino. Nous **devons** placer une résistance d'environ 4,7K Ω entre la broche digitale et la terre (GND). Cette dernière nous garantit que la broche sera à LOW quand l'interrupteur est ouvert.

² Cet algorithme peut être retrouvé ici : <http://www.ganssle.com/debouncing-pt2.htm>

Dans un sketch, on initialise habituellement la broche de la manière suivante :

```
pinMode(2, INPUT)
```

En mode d'excursion basse, quand le contact est fermé, digitalRead() retournera HIGH. Quand le contact est ouvert, il retournera LOW.

Mode excursion basse (PULLDOWN)		Mode excursion haute (PULLUP)	
OUVERT	FERMÉ	FERMÉ	OUVERT
			
			
LOW	HIGH	HIGH	LOW

LA LIBRAIRIE

CRÉER UN BOUTON

Pour pouvoir utiliser cette librairie, il faut d'abord l'ajouter à l'IDE de l'Arduino. Ce site explique bien comment faire : <https://knowledge.parcours-performance.com/librairies-arduino-installer/>

Dans le sketch, on doit mentionner que l'on désire l'utiliser. La ligne suivante doit apparaître au début du sketch, avant la section setup() :

```
#include <EdgeDebounce.h>
```

Par la suite, vous pouvez donner un nom à la broche à laquelle est attaché l'interrupteur. (Il n'est pas nécessaire de le faire, mais cela rend le code plus facile à lire par la suite et cela permet de n'avoir qu'un seul changement à faire au sketch si on change de broche en chemin.)

```
#define BROCHE_BOUTON 2
```

Par la suite, nous devons instancier l'interrupteur. On va créer un objet de la classe « EdgeDebounce » qui se nommera « bouton ». De plus on doit mentionner que la broche à laquelle est relié l'objet est « BROCHE_BOUTON ». Enfin, on doit indiquer quel montage nous avons utilisé pour relier l'interrupteur à l'Arduino (soit « PULLUP » ou « PULLDOWN »)

```
EdgeDebounce bouton(BROCHE_BOUTON, PULLUP);
```

Dans la section setup(), nous allons initialiser la broche de l'interrupteur avec :

```
bouton.begin();
```

NOTE : Dans la version 1.1, l'initialisation de la broche était exécutée lors de l'instanciation de l'objet. Heureusement, un participant du forum Arduino m'a indiqué que cette façon de faire n'était pas sûre. On m'a suggéré de créer une méthode **.begin()** qui serait exécutée dans la section setup().

LIRE UN BOUTON

Il est possible de vérifier s'il est activé (**pressed**). Rappelez-vous :

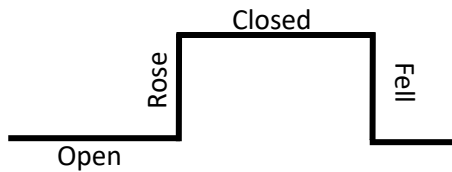
- En mode d'excursion haute, pressed signifie HIGH si l'interrupteur est ouvert et LOW s'il est fermé;
- En mode d'excursion basse, pressed signifie LOW si l'interrupteur est ouvert et HIGH s'il est fermé.

```
if (bouton.pressed() == HIGH) {  
    Votre code ici;  
}
```

Version 1.2 : Cette méthode a été renommée **debounce**. L'ancien nom, **pressed**, est encore disponible pour conserver une compatibilité avec la version antérieure.

```
if (bouton.debounce() == HIGH) {  
    //Votre code ici;  
}
```

LES PHASES DU SIGNAL



Puisque EdgeDebounce a été averti du mode de connexion du bouton (PULLUP ou PULLDOWN) il lui est possible de retourner les phases du signal correctement. Quand le bouton est relâché, il n'y a aucun voltage sur la broche. On dit que l'interrupteur est ouvert (Open). Quand il est pressé, le voltage monte à 5V sur la broche. On dit que l'interrupteur est fermé (Closed). Mais il y a

aussi deux états qui peuvent nous intéresser : Le moment où le signal passe de 0V à 5V, le front montant (Rose) et le moment où le signal passe de 5V à 0V, le front descendant (Fell).

Version 1.1 Nous pouvons demander si le bouton est fermé (closed) :

```
if (bouton.Open()) {  
    //Faire quelque chose  
}
```

Version 1.2 Il est aussi possible de savoir s'il est ouvert (open) :

```
if (!bouton.Open()) {  
    //Faire quelque chose  
}
```

Version 1.2. Il est maintenant possible de savoir si :

Le signal est dans le front montant :

```
if (bouton.rose()) {  
    //Faire quelque chose  
}
```

Le signal est dans un front descendant :

```
if (bouton.fell()) {  
    //Faire quelque chose  
}
```

PRENDRE NOTE QUE : La lecture d'un front montant ou descendant exige que le bouton s'y consacre exclusivement. Il n'y a qu'une occurrence d'un front par cycle et ce cycle peut comporter plusieurs dizaines de milliers de lectures.

Si nous devons absolument vérifier le front montant **et** le front descendant, écrire :

```
bouton.update();  
if (bouton.getRose()) {  
    //Faire quelque chose  
}  
If (bouton.getFell()) {  
    //Faire quelque chose  
}
```

UNE DERNIÈRE CHOSE.

Il est possible d'ajuster la sensibilité du stabilisateur (debouncer) en modifiant la taille des rafales de lectures entre 1 et 32.

Nous pouvons utiliser le sketch exemple « TestSensitivity.ino » pour repérer plus facilement quelle sensibilité est la meilleure pour notre interrupteur ou notre application.

Dans la section setup() de notre sketch, nous mettrons :

```
bouton.setSensitivity(8);
```

On peut aussi obtenir quelle est la sensibilité actuelle avec :

```
byte laSensibilite = bouton.getSensitivity();
```

EXEMPLE DE LA MÉTHODE fell()

Si on veut détecter un click, la méthode classique est :

```
if (digitalRead(pin, HIGH) {  
    while (digitalRead(pin, HIGH) {;} //Attendre tant que la broche reste HIGH  
    //Faire quelque chose  
}
```

Avec la méthode fell() :

```
if (clic.fell()) {    //clic ne peut appeler que .fell() (Il doit se concentrer)  
    //Faire quelque chose  
}
```

EXEMPLE COMPLET

```
#include <EdgeDebounce.h>

#define BROCHE_BOUTON 2
#define BROCHE_DEL 13

//Créer une instance de l'objet Debounce et la nommer bouton
//bouton est relié à BROCHE_BOUTON et est en mode PULLUP
EdgeDebounce bouton(BROCHE_BOUTON, PULLUP);

void setup() {
    bouton.begin();
    pinMode(BROCHE_DEL, OUTPUT);
}

//LA DEL intégrée à la broche 13 s'allume quand l'interrupteur est fermé
void loop() {
    if (bouton.closed()) {
        digitalWrite(BROCHE_DEL, HIGH);
    }
    else {
        digitalWrite(BROCHE_DEL, LOW);
    }
}
```

J'espère sincèrement que la Librairie EdgeDebounce vous sera utile dans vos projets
Jacques Bellavance